下载APP

06 | 聚类分析：如何用RFM给电商用户做价值分组画像？

2021-09-10 黄佳

《零基础实战机器学习》 课程介绍 >

讲述：黄佳
时长 20:42 大小 18.97M

你好，我是黄佳。欢迎来到零基础实战机器学习。

在上一讲中，我们从一份互联网电商“易速鲜花”的历史订单数据集中，求出了每一个用户的 R、F、M 值。你可能会问，从这些值中，我们又能看出什么有价值的信息呢？

别着急，在这一讲中，我们继续往前走，看看如何从这些枯燥且不容易观察的数据中，得到更为清晰的用户分组画像。通过这节课，我希望你能理解聚类算法的原理和最优化过程，这可以帮你建立起针对问题选择算法的直觉。

怎么给用户分组比较合适？

这是我们在上节课中得出的用户层级表，表中有每位用户的 R、F、M 值。

	用户码	R值	F值	M值
0	14681	70	7	498.95
1	14682	187	2	52.00
2	14684	25	390	1201.51
3	14687	106	15	628.38
4	14688	7	324	12130.60

这里，我们希望看看 R 值、F 值和 M 值的分布情况，以便为用户分组作出指导。代码是接着上一讲的基础上继续构建，我就不全部贴上来了，完整的代码和数据集请你从这里下载。

复制代码

```
df_user['R值'].plot(kind='hist', bins=20, title = '新进度分布直方图') #R值直方图
```

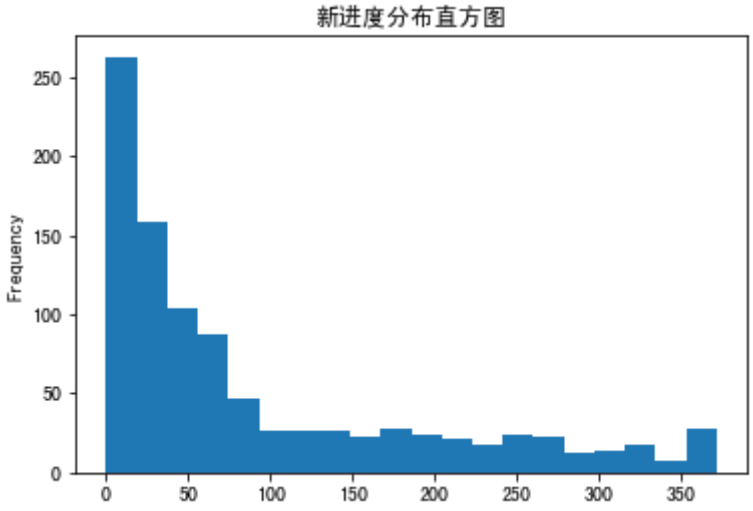
复制代码

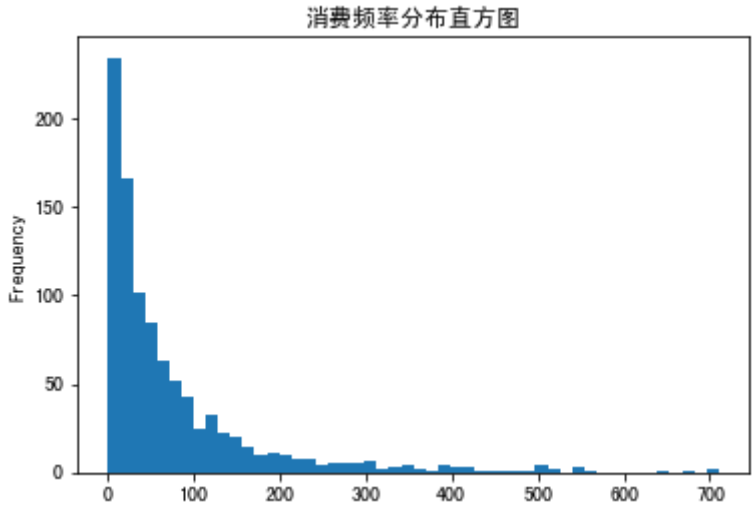
```
df_user.query('F值 < 800')['F值'].plot(kind='hist', bins=50, title = '消费频率分
```

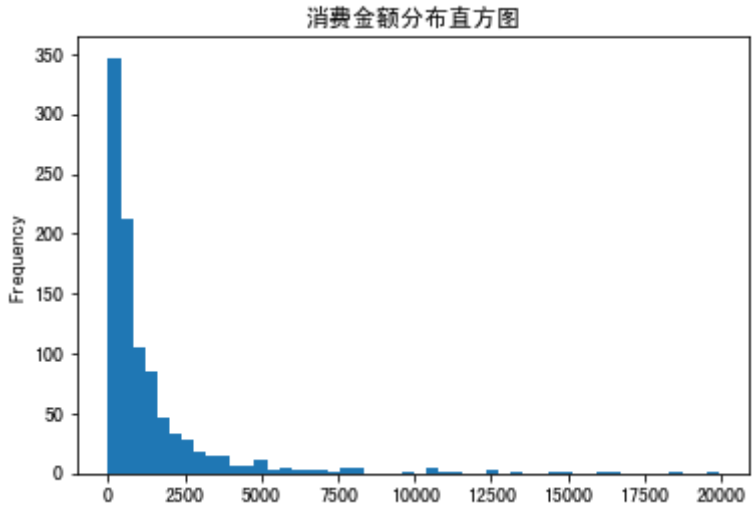
复制代码

```
df_user.query('M值 < 20000')['M值'].plot(kind='hist', bins=50, title = '消费金额
```

分别输出如下结果：







可以看到，我们求出的 R 值、F 值和 M 值的覆盖区间都很大。就拿 R 值来说，有的用户 7 天前购物，R 值为 7；有的用户 70 天前购物，R 值为 70；还有的用户 187 天前购物，R 值为 187。

那现在问题来了，如果说我们的目标是根据 R 值把用户分为几个不同的价值组，那么怎么分组比较合适呢？

其实，这个问题又可以拆分为两个子问题：

1. 分成多少个组比较好？
2. 从哪个值到哪个值归为第一组（比如 0-30 天是一组），从哪个值到哪个值归为第二组（比如 30 天 -70 天是一组）？

对于这两个问题的答案，有人肯定会说，可以凭借经验来人为确定。比如说，把用户分为高、中、低三个组，比如 R 值为 0 到 50 的分为一个组，50 到 150 的分为一组，150 天

以上的归为一组。

这样的人为分组似乎也可以，但它存在一些弊端：首先，分组的准确性完全取决于人的经验，如果分得不准，效果就不好。其次，人为分组是静态的，如果用户情况变化了，我们还是用同样的区间来分组，就不是很合适。

那怎么办呢？我想你已经猜到了，其实该怎么分组，我们说了不算，要数据说了算。所以，要解决这个问题，还是要通过机器学习算法，根据数据的实际情况来动态地确定分组。因为只有这样的模型才是动态的，才能长期投入使用。

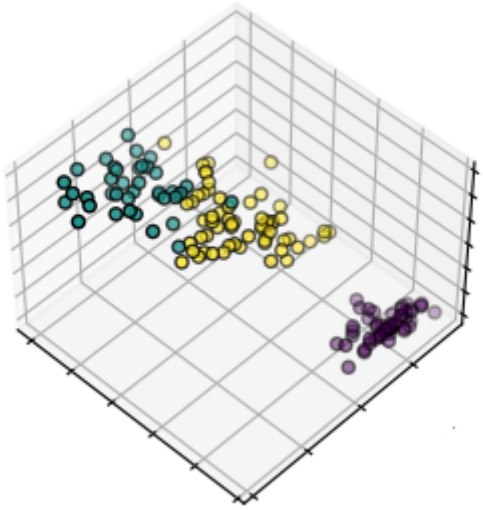
坚定了这一点后，我们考虑一下选什么算法来建立模型。

聚类算法中的 K-Means 算法

首先，我们要搞清楚，给用户做分组画像属于监督学习问题，还是无监督学习问题？我们要通过历史订单数据来给用户分组，这是没有任何已知标签可以做参照的，数据集中并没有一个字段指明用户的价值是“高”还是“低”，所以这显然是一个无监督学习问题。

在无监督学习中，聚类和降维是两种最常见的算法，不过它们应用场景很不一样。聚类我们说过了，主要可以用来做分组；而降维，则是通过数学变换，将原始高维属性空间转变为一个低维“子空间”，它本质上是通过最主要的几个特征维度实现对数据的描述。很显然，我们的问题适合用聚类算法来解决。

聚类算法可以让机器把数据集中的样本按照特征的性质分组，不过它只是帮我们把数据特征彼此邻近的用户聚成一组（这里的组称为聚类的簇）。而这里说的“特征彼此邻近”，指的这些用户的数据特征在坐标系中有更短的向量空间距离。也就是说，**聚类算法是把空间位置相近的特征数据归为同一组。**



聚类算法把空间位置相近的数据归为同一组

不过，请你注意，聚类算法本身并不知道哪一组用户是高价值，哪一组用户是低价值。分完组之后，我们还要根据机器聚类的结果，人为地给这些用户组贴标签，看看哪一组价值高，哪一组价值低。我这里把这种人为贴标签的过程称为“**聚类后概念化**”。等你学完这节课，就能更清楚我为什么要做“聚类后概念化”了。

搞清楚问题适合用聚类算法解决还不够，因为聚类的算法可不止一种，我们还要进一步确定采用哪一个算法。这里我直接选用 K-Means（K- 均值）算法了，因为这个算法不仅简洁，而且效率也高，是我们最常用的聚类算法。像文档归类、欺诈行为检测、用户分组等等这些场景，我们往往都能用到。

说到这里，你也许很疑惑：在监督学习中，模型都是要依赖于标签才能创建出来，这无监督学习怎么就这么聪明，能在没有标签的情况下自动给数据点分组？下面我就带你探寻其中的奥秘。

在 K-Means 算法中，“K”是一个关键。K 代表聚类的簇（也就是组）的个数。比如说，我们想把 M 值作为特征，将用户分成 3 个簇（即高、中、低三个用户组），那这里的 K 值就是 3，并且需要我们人工指定。

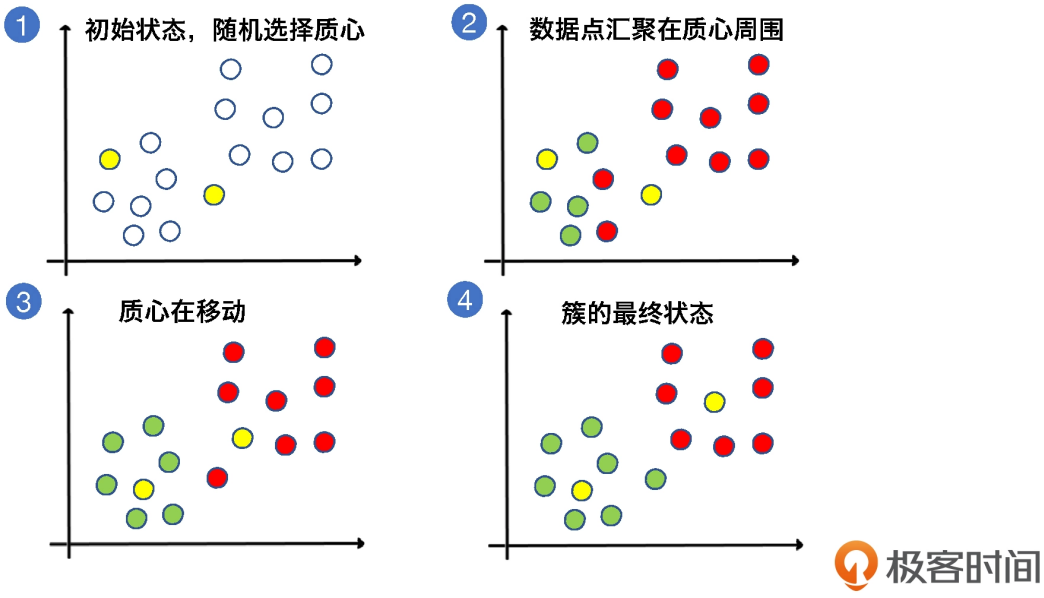
指定 K 的数值后，K-Means 算法会在数据中随机挑选出 K 个数据点，作为簇的质心（centroid），这些质心就是未来每一个簇的中心点，算法会根据其它数据点和它的距离来进行聚类。

挑选出质心后，K-Means 算法会遍历每一个数据点，计算它们与每一个质心的距离（比如欧式距离）。数据点离哪个质心近，就跟哪个质心属于一类。

遍历结束后，每一个质心周围就都聚集了很多数据点，这时候啊，算法会在数据簇中选择更靠近中心的质心，如果原来随机选择的质心不合适，就会让它下岗。

在整个聚类过程中，为了选择出更好的质心，“挑选质心”和“遍历数据点与质心的距离”会不断重复，直到质心的移动变化很小了，或者说固定不变了，那 K-Means 算法就可以停止了。

我用下面的图来帮助你理解质心在聚类过程中逐渐移动到最佳位置，以及簇的形成过程：



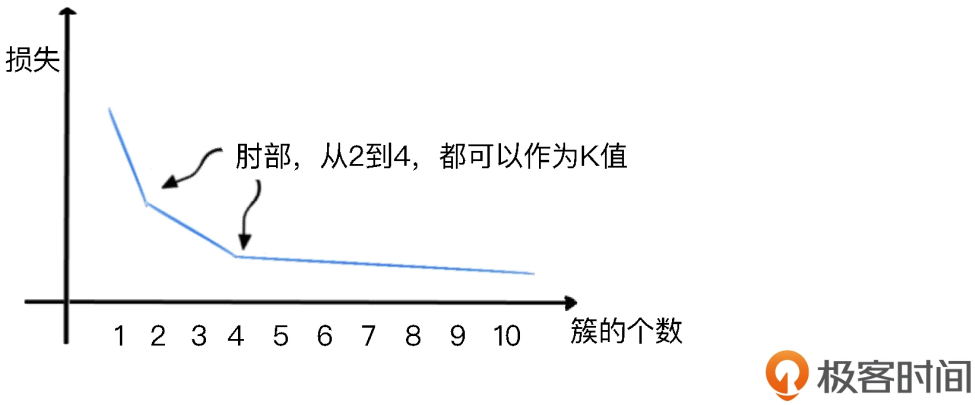
聚类中心（质心）的移动和簇形成的过程

理解了聚类算法的原理，我们继续来思考一个问题：我们前面说 K 值需要人工指定，那怎么在算法的辅助下确定 K 值呢？

手肘法选取 K 值

其实，在事先并不是很确定分成多少组比较合适的情况下，“手肘法”（elbow method）可以帮我们决定，在某一批数据点中，数据分为多少组比较合适。这里我要特别说明一下，尽管我们前面说要把用户分为高、中、低三个价值组，但是 R、F、M 的值却可以分成很多组，并不一定都是 3 组。

手肘法是通过聚类算法的损失值曲线来直观确定簇的数量。损失值曲线，就是以图像的方法绘出，取每一个 K 值时，各个数据点距离质心的平均距离。如下图所示，当 K 取值很小的时候，整体损失很大，也就是说各个数据点距离质心的距离特别大。而随着 K 的增大，损失函数的值会在逐渐收敛之前出现一个拐点。此时的 K 值就是一个比较好的值。



手肘法——确定最佳K值

你看图中，损失随着簇的个数而收敛的曲线大概像个手臂，最佳 K 值的点像是一个手肘，这就是为什么我们会叫它“手肘法”的原因。

下面我们就用代码找出 R 值的手肘点。请你注意，这里我会先定义一个找手肘点的函数，因为后面在对 R 值、F 值和 M 值聚类的过程中，我们都要用到这个函数。

复制代码

```
from sklearn.cluster import KMeans #导入KMeans模块
def show_elbow(df): #定义手肘函数
    distance_list = [] #聚质心的距离（损失）
    K = range(1,9) #K值范围
    for k in K:
        kmeans = KMeans(n_clusters=k, max_iter=100) #创建KMeans模型
        kmeans = kmeans.fit(df) #拟合模型
        distance_list.append(kmeans.inertia_) #创建每个K值的损失
    plt.plot(K, distance_list, 'bx-') #绘图
    plt.xlabel('k') #X轴
    plt.ylabel('距离均方误差') #Y轴
    plt.title('k值手肘图') #标题
```

在这段代码中，核心部分是拟合 kmeans 模型之后，通过 kmeans.inertia_ 计算损失值。损失会随着 K 值的增大而逐渐减小，而那个拐点就是手肘。

然后我们调用下面这个函数，显示 R 值、F 值和 M 值聚类的 K 值手肘图：

```
1 show_elbow(df_user[['R值']]) #显示R值聚类K值手肘图
```

复制代码

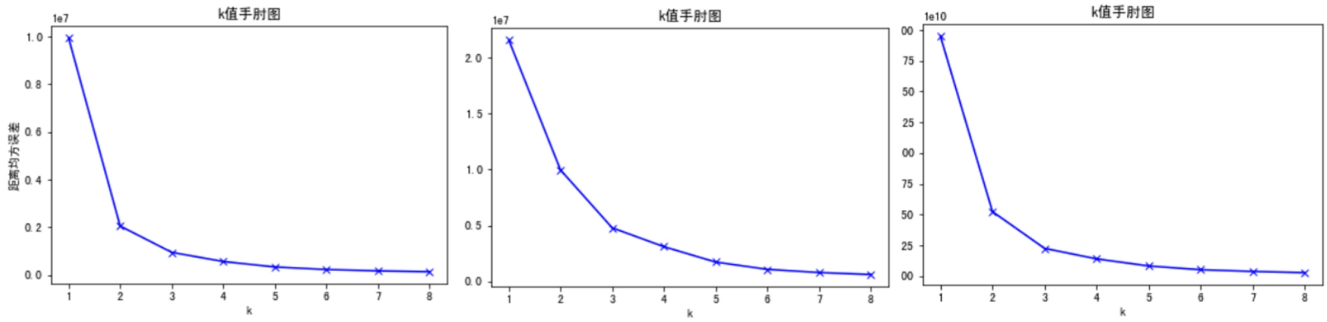
```
1 show_elbow(df_user[['F值']]) #显示F值聚类K值手肘图
```

复制代码

```
1 show_elbow(df_user[['M值']]) #显示M值聚类K值手肘图
```

复制代码

输出如下：



R、F和M值的K值手肘图

可以看到，R、F、M 值的拐点大概都在 2 到 4 之间附近，这就意味着我们把用户分成 2、3、4 个组都行。这里我选择 3 作为 R 值的簇的个数，选择 4 作为 F 值的簇的个数，选择 3 作为 M 值的簇的个数。

那到这里为止呢，我们已经选定好了算法，并确定了 R、F、M 每个特征下簇的个数，也就是 K 值。接下来我们就可以开始创建聚类模型了。

创建和训练模型

前面说了，我们在手肘附近选择 3 作为 R 的 K 值，所以我们创建模型是把 n_clusters 参数，也就是簇的个数指定为 3。这样，聚类算法会把用户的 R 值分为三个层次。对于 F、M，我们也根据对应的 K 值做类似的操作：

```
from sklearn.cluster import KMeans #导入KMeans模块
kmeans_R = KMeans(n_clusters=3) #设定K=3
kmeans_F = KMeans(n_clusters=4) #设定K=4
kmeans_M = KMeans(n_clusters=4) #设定K=4
```

这样，我们就在程序中创建了一个 K-Means 聚类模型。

创建好模型后，我们借助 fit 方法，用 R 值的数据，训练模型。

```
kmeans_R.fit(df_user[['R值']]) #拟合模型
kmeans_F.fit(df_user[['F值']]) #拟合模型
kmeans_M.fit(df_user[['M值']]) #拟合模型
```

我们不是第一次见到 fit 这个方法了，fit，翻译成中文就叫做拟合模型。基本上所有的机器学习模型都是用 fit 语句来进行模型训练的。

使用模型进行聚类，并给用户分组

模型训练好了，现在我们就用它给 R、F、M 值聚类。

1. 给 R、F、M 值聚类

我们先用 kmeans 模型中的 predict 方法给 R 值聚类。“predict”翻译成中文是“预测”，不过作为无监督学习方法，它其实就是使用模型进行聚类，而且，也不需要进一步的评估过程。这也是监督学习和无监督学习不一样的地方。

```
df_user['R值层级'] = kmeans_R.predict(df_user[['R值']]) #通过聚类模型求出R值的层级
df_user.head() #显示头几行数据
```

这段代码的输出如下：

	用户码	R值	R值层级
0	14681	70	1
1	14682	187	0
2	14684	25	1
3	14687	106	0
4	14688	7	1

输出显示，这个聚类结果被附加到了用户层级表中，也就是说在用户层级表中的“用户码”、“R 值”字段后面出现了“R 值层级”这个字段，也就是将 R 值聚类后各个簇的号码。

下面我们用 groupby 语句来看看 0、1、2 这几个簇的用户基本统计数据：

复制代码

```
df_user.groupby('R值层级')['R值'].describe() #R值层级分组统计信息
```

这段代码的输出如下：

	count	mean	std	min	25%	50%	75%	max
RecencyCluster								
0	670.0	31.225373	25.165371	0.0	9.0	24.0	49.0	94.0
1	139.0	295.978417	44.624921	231.0	254.5	289.0	331.0	372.0
2	177.0	157.141243	36.992127	95.0	126.0	157.0	188.0	225.0

这里有一个奇怪的现象，不知道你有没有观察到？

如果你注意看 0、1 和 2 这三个簇，也就是三个组，就会发现形成的簇没有顺序。你看，0 群的用户最多 670 个人，均值显示他们平均购物间隔是 31 天，上次购物距今是 0 天到 94 天，这是相对频繁的购物用户群。

1 群的用户平均购物间隔为 295 天，上次购物距现在是 231 天到 372 天，这是在休眠中的用户；而 2 群的用户平均购货间隔则变成了 157 天，介于两者之间，他们上次购物距今是从 95 天到 225 天。你会发现这个从 0 到 2 的顺序既不是升序，也不是降序。

这其实是聚类这种算法本身的问题。聚类，作为一种无监督学习算法，是不知道顺序的重要性的，它只是盲目地把用户分群（按照其空间距离的临近性），而不管每个群的具体意义，因此也就没有排序的功能。这也就是我前面说的“聚类后概念化”的具体意思。聚类并不知道那组人的价值高低，所以也就无法确定顺序，需要我们人为来排序。

2. 为聚类的层级做排序

那么，下面我们就用一段代码，把聚类的结果做一个排序，让 0、1、2 这三个组体现出价值的高低。这段代码稍微有点长，不过我给出了详细的注释，你可以看一下：

复制代码

```
#定义一个order_cluster函数为聚类排序
def order_cluster(cluster_name, target_name,df,ascending=False):
    new_cluster_name = 'new_' + cluster_name #新的聚类名称
    df_new = df.groupby(cluster_name)[target_name].mean().reset_index() #按聚类
    df_new = df_new.sort_values(by=target_name,ascending=ascending).reset_inde
    df_new['index'] = df_new.index #创建索引字段
    df_new = pd.merge(df,df_new[[cluster_name,'index']], on=cluster_name) #基于
    df_new = df_new.drop([cluster_name],axis=1) #删除聚类名称
    df_new = df_new.rename(columns={"index":cluster_name}) #将索引字段重命名为聚类
    return df_new #返回排序后的df_new对象
```

在上述代码中，为聚类做排序的是 order_cluster 函数。那么接下来，我们再调用这个 order_cluster 函数，把用户表重新排序。我们知道，消费天数间隔的均值越小，用户的价值就越高，所以我们在这里采用降序，也就是把 ascending 参数设为 False：

复制代码

```
df_user = order_cluster('R值层级', 'R值', df_user, False) #调用簇排序函数
df_user = df_user.sort_values(by='用户码',ascending=True).reset_index(drop=True
df_user.head() #显示头几行数据
```

此时，各用户的层级值就发生了变化，比如用户 14688 的簇编号从 1 变成了 2，因为这个用户 7 天前曾经购物，其 R 值相对偏低，放在高分的 2 层级是合适的。其实，上面的代码中，我们并没有改变用户的分组，而只是改变了每一个簇的编号，这样层级关系就能体现出来了。

下面我们重新显示各个层级的信息：

复制代码

```
df_user.groupby('R值层级')['R值'].describe() #R值层级分组统计信息
```

输出如下：

R值层级	count	mean	std	min	25%	50%	75%	max
0	138.0	298.094203	45.436550	231.0	255.25	292.5	334.50	372.0
1	178.0	157.162921	37.340870	95.0	126.00	156.5	188.75	225.0
2	664.0	32.088855	25.141763	0.0	10.00	25.0	50.00	94.0

你会看到，此时各个簇已经形成了次序。0 层级的用户，平均新近度是 298 天，1 层级的用户，平均新近度是 157 天，而 R 值最高的用户组（2 层级），平均新近度仅有 32 天。这说明用户上一次消费距今的天数越少，其 R 值的价值越高。

R 值聚类做好后，我们按照同样的方法可以根据用户购买频率给 F 值做聚类，并用刚才定义的 order_cluster 函数为聚类之后的簇进行排序，确定层级。因为消费次数越多，价值越高，所以我们把 order_cluster 函数的 ascending 参数设定为 True，也就是升序：

复制代码

```
df_user['F值层级'] = kmeans_F.predict(df_user[['F值']]) #通过聚类模型求出F值的层级
df_user = order_cluster('F值层级', 'F值',df_user,True) #调用簇排序函数
df_user.groupby('F值层级')['F值'].describe() #F值层级分组统计信息
```

输出如下：

F值层级	count	mean	std	min	25%	50%	75%	max
0	718.0	32.399721	24.333294	1.0	12.00	27.0	49.00	92.0
1	202.0	153.193069	49.685452	93.0	114.25	139.5	187.25	278.0
2	53.0	416.584906	111.990974	286.0	327.00	388.0	498.00	710.0
3	7.0	1295.285714	516.333456	899.0	1013.50	1119.0	1321.50	2379.0

上图显示，经过了排序的层级中，0 级用户的购买频率均值为 32 次，1 级的用户消费频率均值为 153 次，2 级用户消费频率均值达到 416 次，而 2 级用户的消费频率均值高达

1295 次，不过这个簇中只有 7 个用户。

还是一样，我们重新为用户层级表排序，并显示 df_user 对象，也就是用户层级表的当前状态。

```
df_user = df_user.sort_values(by='用户码',ascending=True).reset_index(drop=True
df_user.head()
```

输出如下：

	用户码	R值	R值层级	F值	F值层级
0	14681	70	2	7	0
1	14682	187	1	2	0
2	14684	25	2	421	2
3	14687	106	1	15	0
4	14688	7	2	327	2

最后，我们依葫芦画瓢，给 M 值做聚类，并且对聚类的结果做排序，分出层级。因为代码和 R 值、F 值聚类十分相似，我就直接给出所有代码，不再说明了。

```
df_user['M值层级'] = kmeans_M.predict(df_user[['M值']]) #通过聚类模型求出M值的层级
df_user = order_cluster('M值层级', 'M值',df_user,True) #调用簇排序函数
df_user.groupby('M值层级')['M值'].describe() #M值层级分组统计信息
df_user = df_user.sort_values(by='用户码',ascending=True).reset_index(drop=True
df_user.head() #显示头几行数据
```

最终结果：

	用户码	R值	R值层级	F值	F值层级	M值	M值层级
0	14681	70	2	7	0	498.95	0
1	14682	187	1	2	0	52.00	0
2	14684	25	2	421	2	1236.28	0
3	14687	106	1	15	0	628.38	0
4	14688	7	2	327	2	5630.87	1

好，那到这里为止，R、F、M 的聚类工作就全部完成了，并且我们还划分了层级！在当前的用户层级表中，已经包含了这三个维度的层级，最终的用户分层就可以以此为基础来确定了。

为用户整体分组画像

我们这里采用简单叠加的方法把 R、F、M 三个层级的值相加，用相加后得到的值，作为总体价值，来给用户进行最终的分层。当然了，如果你对其中某一个指标看得比较重，也可以加权之后再相加。

具体来讲，我们用下面的代码来创建相加之后的层级，即总分字段。

复制代码

```
df_user['总分'] = df_user['R值层级'] + df_user['F值层级'] + df_user['M值层级'] #求
```

因为 R 值有 3 个层级（0，1，2），F 值有 4 个层级（0，1，2，3），M 值有 4 个层级（0，1，2，3），我们把三个维度的值相加，那每一个用户的得分有可能是 0 到 8 当中的某一个值，也就是说出现了 9 个层次。

我这里就按照下面的规则，来确定用户最终的价值分层。当然了，你也可以尝试用其它的阈值来确定你的价值分层。

0-2 分，低价值用户

3-4 分，中价值用户

5-8 分，高价值用户

什么意思呢？举例来说，就是如果一个用户在 R 值拿到了 2 分，在新近度这个维度为高价值用户，但是在消费频率和消费金额这两个维度都只拿到 0 分，那么最后得分就为 2，总体只能评为低价值用户。

下面这段代码便是根据总分，来确定出每一个用户的总体价值的。

复制代码

```
#在df_user对象中添加总体价值这个字段
```

```
df_user.loc[(df_user['总分']<=2) & (df_user['总分']>=0), '总体价值'] = '低价值'
df_user.loc[(df_user['总分']<=4) & (df_user['总分']>=3), '总体价值'] = '中价值'
df_user.loc[(df_user['总分']<=8) & (df_user['总分']>=5), '总体价值'] = '高价值'
df_user #显示df_user
```

我们再次输出 df_user 对象，看一看最终的用户层级表：

	用户码	R值	R值层级	F值	F值层级	M值	M值层级	总分	总体价值
0	14681	70	2	7	0	498.95	0	2	低价值
1	14682	187	1	2	0	52.00	0	1	低价值
2	14684	25	2	421	2	1236.28	0	4	中价值
3	14687	106	1	15	0	628.38	0	1	低价值
4	14688	7	2	327	2	5630.87	1	5	高价值
...	...	...	...	...	...	...	...	...	...
975	16015	3	2	182	1	705.39	0	3	中价值
976	16016	2	2	235	1	1508.76	0	3	中价值
977	16017	46	2	32	0	211.88	0	2	低价值
978	16018	38	2	28	0	408.90	0	2	低价值
979	16019	46	2	160	1	3786.24	1	4	中价值

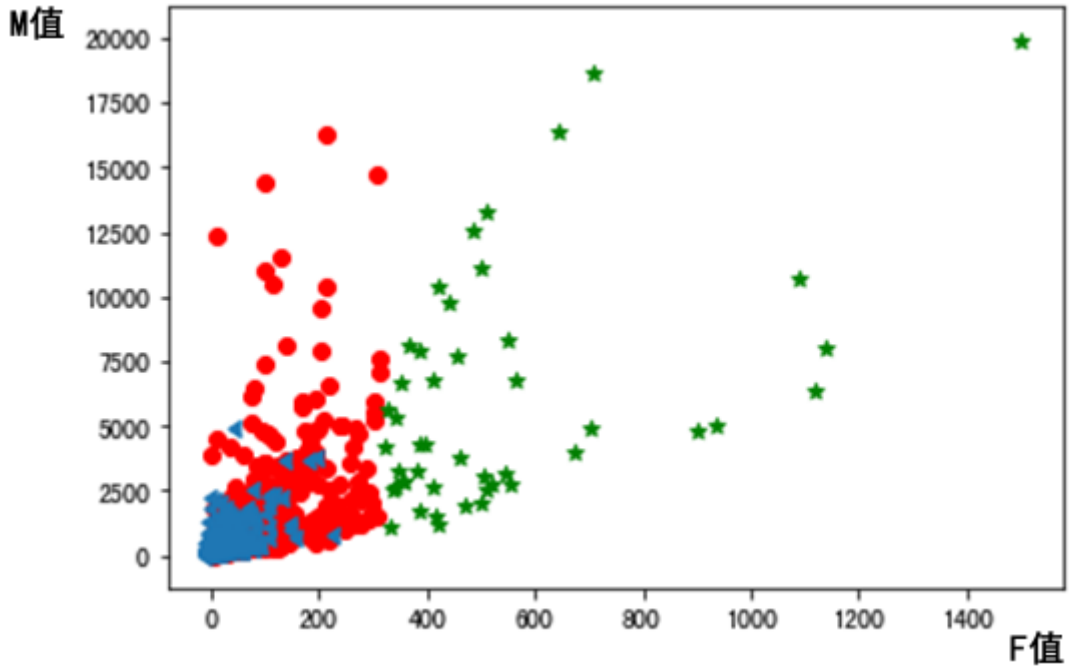
此时，980 个用户的 R、F、M 层级，还有总体价值的层级都非常清楚了。对于每一个用户，我们都可以迅速定位到他的价值。那到这里，我们就成功地完成了为“易速鲜花”公司做用户价值分组的工作。

现在，有了用户的价值分组标签，我们就可以做很多进一步的分析，比如说选取 R、F、M 中任意两个维度，并把高、中、低价值用户的散点图进行呈现：

复制代码

```
#显示高、中、低价值组分布散点图（F值与M值）
plt.scatter(df_user.query("总体价值 == '高价值'")['F值'],
                    df_user.query("总体价值 == '高价值'")['M值'],c='g',marker='*')
plt.scatter(df_user.query("总体价值 == '中价值'")['F值'],
                    df_user.query("总体价值 == '中价值'")['M值'],marker=8)
plt.scatter(df_user.query("总体价值 == '低价值'")['F值'],
                    df_user.query("总体价值 == '低价值'")['M值'],c='r')
各价值组的用户分布散点图如下图所示：

```



借此，我们可以发现，高价值用户（绿色五星）覆盖在消费频率较高的区域，和 F 值相关度高。而在总消费金额大于 5000 元的用户中，中高价值的用户（绿色五星和红色圆点）都有。

当然，作为运营部门的一员，你还可以通过对新老用户的价值分组，制定出更有针对性的获客、营销、推广等运营方案。

总结一下

好啦，到这里，我们就成功闯过了“获客”这个关卡。现在来回顾一下这一讲中的重点内容。

在这一讲中， 我们理解了聚类算法的原理，学会了如何用手肘法选择 K 值，以及如何用 K- 均值算法来给 R 值、F 值和 M 值做聚类。

我要再次强调的是，聚类只是负责把空间距离相近的数据点分成不同的簇，它并不知道每一个簇代表的具体含义。比如说，我们用 K-Means 算法给 R 值分成三个簇，这并不表示 0 比 1 价值低，1 比 2 价值低。此时的 0、1、2 都是聚类随机分配的编号，具体分组排序的工作我们还要单独去完成。

在 K 值的选择方面呢，手肘法可以帮我们直观地显示出聚类过程中整体损失的“拐点”，我们可以在拐点或者拐点附近选择 K 值，确定把数据分成多少个“簇”（也就是多少个

组）。

最后，我们还讲到用 K- 均值算法来给 R 值做聚类，这也非常简单，就是创建模型、拟合模型、用模型进行聚类，这些过程加一块也就是几行代码的事儿，你不用有负担。

掌握了上述这些内容，你就可以用 K-Means 算法这种无监督学习算法给任何数据集做聚类，来解决其它类似的问题了，比如根据学生的考试成绩，为学生分组聚类等等。

思考题

这节课就到这里了，我给你留两个思考题：

1. 对于 K-Means 算法，X 特征数据集的输入，可以不止一个维度。为了给 R、F、M 分别分层，在这节课中我给这三者单独做了聚类。你能不能试着把 R、F、M 三个特征同时输入 K-Means 算法，为用户整体做聚类呢？此外，你还能不能想到些其它的为用户分组画像的方法呢？
2. 聚类算法的应用场景其实很广，包括给图像的颜色簇量化分组、给文本分组等等，在你的工作和生活中，你还能够想到，或者曾用过哪些可以通过聚类解决的问题？请你分享一下。

欢迎你在留言区和我分享你的观点，如果你认为这节课的内容有收获，也欢迎把它分享给你的朋友，我们下一讲再见！

极客时间

本讲重点

聚类算法结合 RMF 分析为用户分组的步骤：

1. 选取 R 值、F 值和 M 值作为独立特征；
2. 用手肘法确定 R、F、M 的层级个数；
3. 以 R 值为特征，进行聚类，确定层级；
4. 以 F 值为特征，进行聚类，确定层级；
5. 以 M 值为特征，进行聚类，确定层级；
6. 把各个层级进行统合，确定用户最终价值。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 4 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 05 | 数据探索：怎样从数据中找到用户的RFM值？

下一篇 07 | 回归分析：怎样用模型预测用户的生命周期价值？

精选留言 (4)

写留言

吴悦
2021-09-10

多维数据输入时怎么做 无量纲化好呀

展开

作者回复: 同学们的问题总是会引导我剧透后面的内容。。。

在sklearn中，同学所说的无量纲化英文被称为特征的缩放(Scaler)，把大的数值压缩到小的范围，或者改变数值的分布状态等。具体的方法和工具很多，包括标准化，归一化，居中，规范化等等不同的手段（这些手段的名词也容易混淆）：

1）StandardScaler工具，标准化缩放，是对数据特征分布的转换，目标是使其符合正态分布(均值为0，方差为1)。对于某些模型，如果数据特征不符合正态分布的话，就影响机器学习效率。

2）MinMaxScaler工具，是把特征的值压缩到给定的最小值和最大值之间，通常在0和1之间，有负值的话就是-1到1，因此也叫归一化。归一化不会改变数据的分布状态。在sklearn中，通过MinMaxScaler进行标准化缩放。某些模型，比如神经网络，就非常喜欢归一化之后的特征数据。

3）RobustScaler工具，基于百分位数的缩放，能消除这个过程消除了数据中的离群值的影响，但转换后的特征值的结果范围比之前的更大，所以RobustScaler之后，通常还结合归一化缩放器一起使用，再把数据压缩一下。

4）Normalizer工具，规范化缩放，则是将样本缩放为具有单位范数的过程，这可能仅适用于某些各个维度都是One-Hot编码的数据集，应用场景较少。

不同数据集适用不同工具。

这些我们以后要讲。

4

在路上

2021-09-10

佳哥好，今天是教师节，祝你节日快乐。我看了下kmeans的API，fit函数支持多维数组，应该可以把RFM三个维度的值同时传给fit进行拟合，而不是把RFM三个维度的值降维到一个值进行拟合，通过predict函数就可能直接得到聚类后的层级，从predict函数的名称可以得知，它不仅能计算训练集的聚类层级，也能预测新数据的聚类层级。
我从事游戏行业，策划会根据玩家充值金额来定义大中小R，充值区间的定义全凭经验，...

展开

作者回复: 谢谢精辟的分享，皆是宝贵的经验之谈，对我亦有启发！三人行，必有我师！大家也是我的老师，我这儿也祝愿大家工作生活愉快。

对，文本部分，我这儿需要按照表格中跑出来的值进行调整一下，把二者对应上。谢谢你的细心！！

3

Jove

2021-09-14

1、绘制三维图
df_user['三维价值'] = KMeans(n_clusters=3).fit(df_user[['R值', 'F值', 'M值']]).predict(df_user[['R值', 'F值', 'M值']])

ax = plt.subplot(111, projection='3d')...

展开

作者回复: 这……这……这简直是太棒了。三维RFM图看起来更直观。漂亮。

感谢Jove同学对于信贷场景的分享！

2

蝶舞清风寒

2021-09-16

R、F、M 三个特征同时输入 K-Means 算法，为用户整体做聚类程序代码如下，是否正确呢？

```
train_x=df_user[["R值","F值","M值"]]#将RFM数值转变为训练数据
# 规范化到[0,1]空间
min_max_scaler=preprocessing.MinMaxScaler()...
```

展开

作者回复: 没问题可以的，整体聚类之后结果如何，是否如实反映出用户价值分组情况呢。